SYNTAX
שפה מייצרת חשיבה

Lesson 4 – Loops

הנושאים הנלמדים:

לולאת for, לולאת while

לולאת while

- לולאת while - ביצוע חוזר כל עוד התנאי מתקיים.
- לפני ביצוע הבלוק נבדק התנאי, אם ערכו true מתבצע שוב ושוב עד שערכו יהיה false.

```
while  ([condition])
{
  //this block will be executed while true
}
```

דוגמה – ניחוש מספר

```
final int WINNING_NUMBER = 9;

Scanner scanner = new Scanner(System.in);

System.out.println("Please guess the winning number");
int guess = scanner.nextInt();

while (guess!=WINNING_NUMBER) {
    System.out.println("oops! try again");
    guess = scanner.nextInt();
}

System.out.println("Well done!!!");
```

המלה השמורה final
מציינת כי המשתנה סופי.
כלומר לאחר איתחולו
הראשוני ערכו קבוע והוא
לא ניתן לשינוי

דוגמה – מספר מינימלי

```
// This program ask the user for numbers until -1 is entered and will show the min number
Scanner scanner = new Scanner(System.in);

int number,min;

System.out.println("Please enter a number or -1 to finish");
min = number = scanner.nextInt();// the first number is the biggest until now

while (number != -1) {// while numbers are entered

    System.out.println("Please enter a number or -1 to finish");
    number = scanner.nextInt();

    if(number<min) // if we found a smaller number then min, save it in min
        min = number;

}

System.out.println("The smallest number is " + min);
```

הדפסת המספר ההפוך(בכל גודל!)

// This program ask the user for a number and print the reverse number (any length!)

```
Scanner scanner = new Scanner(System.in);
int result = 0;  // this will be the reverse number
System.out.println("Please enter a number");
int number = scanner.nextInt();

while (number != 0){ // as long as there are digits – not 0
    int digit = number % 10; // getting the current digit
    number /= 10;  // dividing by 10 removes the last digit
    result = result*10 + digit; // "moving" the last digit to the left and
                                   adding the new one
}

System.out.println("The reverse number is " +result);
```

Input: 43726 Output: 62734

לולאות – זהירות!

```
int num = 1;
while (num>0) {
    num++;
    System.out.println(num);
}
```

נשים לב שהתנאי שבתוך הלולאה תמיד true ולכן הלולאה תמשיך לנצח – לולאה אינסופית.

דוגמה אחרונה - סכום

// This example will ask the user for 10 number and show the sum of them

```
Scanner scanner = new Scanner(System.in);
int counter = 1;

int sum = 0; // initializing the sum to 0 is important cause we add to the old value
while (counter <= 10) {
    System.out.println("Please enter number " + counter);
    int input = scanner.nextInt();
    sum += input;
    counter++;  // counter will hold the number of iterations
}

System.out.println("The sum of the numbers is " + sum);
```

מה ההבדל בין הדוגמה הזו לאחרות ?
רמז, תחשבו על מספר הריצות.

לולאת for

לולאת for עושה שימוש מובנה באינדקס (אחד או יותר) אשר מאותחל על ידי הלולאה, נבדק לפני הכניסה אליה ומעודכן בסופה. האינדקס הזה מונה את מספר האיטרציות (ריצות).

המבנה של הלולאה :

for([איתחול] ; [תנאי] ; [עדכון]) {
//block to be executed
}

דוגמה - הדפסת שלום עולם 20 פעם:

```
for(int i=0;i<20;i++) {
    System.out.println("Hello world!");
}
```

לולאת for - כללים

- בשלב הראשון - איתחול המשתנה ובדיקת התנאי.

אם אמת:

- מתבצעות ההוראות בבלוק הלולאה.
- עדכון המשתנה בסיום.
- לפני הכניסה הבאה נבדק הערך, אם אמת

אם שקר:

- מסתיים ביצוע הלולאה ונמשכת התכנית.

דוגמה נוספת

- הדפסת המספרים הזוגיים בין 0 ל 100

```
for(int i=0;i<=100;i+=2) {
    System.out.println(i);
}
```

לולאת for - תוספות

- ניתן לאתחל ולקדם יותר ממשתנה אחד

```
for(int i=0,j=100;i<=100 && j>=100;i++,j--) {

    System.out.println(i + j);
}
```

- אף אחד מחלקי הלולאה אינו חובה אך ההפרדה באמצעות נקודה פסיק היא חובה

```
for(;;) {
    System.out.println("hello");
}
```

לולאה אינסופית

שיפור דוגמת הסכום

// improvement of last example – sum of 10 numbers – iteration's number is known

```
Scanner scanner = new Scanner(System.in);
int sum = 0;
for(int i=0;i<10;i++) {
    System.out.println("Please enter number " + (i+1));
    int num = scanner.nextInt();
    sum += num;
}
System.out.println("The result is " + sum);
```

- מה חסכנו כאן? מה הרווחנו?
- מדוע פתרון ה for עדיף על פתרון ה while הקודם ?
- האם תמיד ניתן להחליף לולאת while ב for ?

סכום של מספרים (המשך)

// sum of numbers – the user enter how many numbers – iteration's number is known at run time

```
Scanner scanner = new Scanner(System.in);
System.out.println("How many  numbers to sum");
int amount  = scanner.nextInt();

int sum = 0;
for(int i=0;i<amount;i++) {
    System.out.println("Please enter number " + (i+1));
    int num = scanner.nextInt();
    sum += num;
}
System.out.println("The result is " + sum);
```

נשים לב שכאן כמות המספרים לא היתה ידועה מראש
בזמן הכתיבה אך היא כן היתה מוגדרת מראש!

מתי for ומתי while ?

- כל מה שנעשה עם while נוכל גם לעשות עם for וההפך.
- בכלזאת, מתי נעדיף for ומתי while ?

- for - מספר ידוע של פעמים (המספר יכול להקבע גם בזמן ריצת התוכנית).
- while - מספר הפעמים אינו ידוע ותלוי בתנאי כלשהו.

דוגמה – בדיקת ראשונית

- מספר ראשוני - מתחלק רק בעצמו ובאחד.
- צריך לחלק אותו בכל המספרים שקטנים ממנו (או עד שורש המספר).
- אם מספר המחלקים כבר עבר את השתיים האם יש טעם להמשיך לבדוק?

דוגמה – בדיקת ראשוניות

```
Scanner scanner = new Scanner(System.in);
System.out.println("Please enter a number");

int number  = scanner.nextInt();
int dividers = 0; //counts the number of dividers

for(int i=1;i<=number && dividers<=2;i++) {
    if(number % i == 0) //if the number divide by i
        dividers++;
}

if(dividers<=2)
    System.out.println(number + " is prime!");
```

מה חסכנו כאן?

לולאות מקוננות

- גוף הלולאה הוא לולאה נוספת.
- לשים לב לזמני ריצה, שכן מספר האיטרציות הוא כגודל הלולאה הפנימית X גודל החיצונית
- למשל הדפסת לוח הכפל:

```
for ( int i = 1; i <= 10; i++ ) {
    for ( int j = 1; j <= 10; j++ )
        System.out.print(i*j + "\t");
    System.out.println();
}
```

this line will be preformed 10 times for each i

\t add tab space - pre detramined spaces – for nice apearance

לולאות מקוננות - ראשוניים

ניתן לשכלל את הדוגמה הקודמת וכעת נרצה לקבל מספר ולהדפיס את כל הראשוניים הקטנים ממנו

```
Scanner scanner = new Scanner(System.in);
System.out.println("Please enter a number");
int number  = scanner.nextInt();

for(int i=1; i<number; i++) {

    int dividers = 0;

    for (int j = 1; j <= i && dividers <= 2; j++) {
        if (i % j == 0)
            dividers++;
    }

    if (dividers <= 2)
        System.out.println(i);
}
```

שבירת המבנה - break & continue

- פקודת break גורמת ליציאה מיידית מהמבנה
- ב switch זה שימושי מאוד, ב for או while - לא מומלץ!
- תנאי העצירה עדיף בראש הלולאה – בולט וברור.
- continue - לדלג לסוף הבלוק ולאיטרציה הבאה.

דוגמה ל continue הדפסת אי-זוגיים

```
for(int i=0;i<100;i++) {

    if(i%2 == 0) continue;
    System.out.println(i);
}
```

- האם אפשר היה לעשות זאת בדרכים אחרות ?
- כמה ?
- תשתמשו בהן. עדיף.

תוספת – לולאת do - while

- while - יכולה לא להתבצע אפילו פעם אחת.
- do - while הריצה הראשונה מתבצעת תמיד והתנאי נבדק בסוף – נמשיך כל עוד אמת.

```
do {

} while([condition])
```

דוגמה

```
Scanner scanner = new Scanner(System.in);
int number;
do {
    System.out.println("Please enter positive number");
    number = scanner.nextInt();
} while (number<0);
```

- הלולאה תמשיך להתבצע כל עוד המשתמש יכניס לנו מספר שלילי.
- מבנה זה מצויין לבדיקת קלט.
- נשים לב שהמשתנה הנבדק חייב להיות מוגדר מחוץ ללולאה.